Daemons, Deployment and Datacentres

Andrew Godwin
@andrewgodwin

Who am I?

- Django core developer
- South author
- Cofounder of ep.io

What's ep.io?

- Hosts Python sites/daemons
- Technically language-independent
- Supports multiple kinds of database
- Mainly hosted in the UK on our own hardware

What I'll Cover

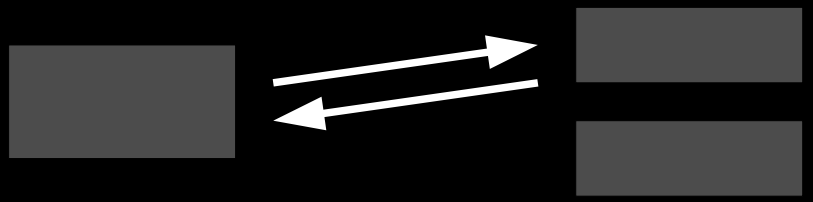
- Our architecture
  - ZeroMQ and redundancy
  - Eventlet everywhere
  - The upload process
  - The joy of networks
- General Challenges
  - "The Stack"
  - Backups and replication
  - Sensible architecture

ZeroMQ & Redundancy

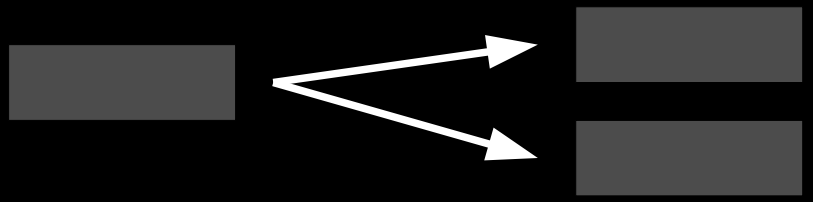
ZeroMQ

- Most importantly, not a message queue
- Advanced sockets, with multiple endpoints
- Has both deliver-to-single-consumer, and deliver-to-all-consumers.
- Uses TCP (or other things) as a transport.

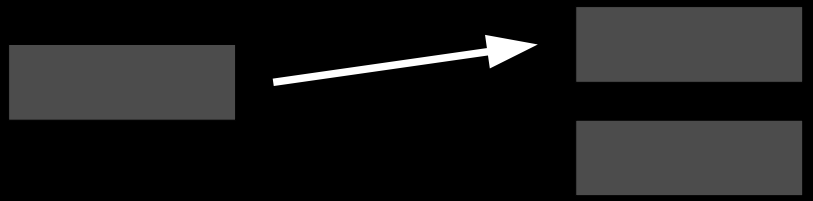
Socket Types



REQ / REP



PUB / SUB



PUSH / PULL

Redundancy

- Our internal rule is that there must be at least two of everything inside ep.io.
- Not quite true yet, but getting very close.
- Even our ”find the servers running X” service is doubly redundant.

Example

```
# Make and connect the socket
sock = ctx.socket(zmq.REQ)
for endpoint in self.config.query_addresses():
     sock.connect(endpoint)
# Construct the message
payload = json.dumps({"type": type, "extra": extra})
# Send the message
with Timeout(30):
    sock.send(self.sign_message(payload))
    # Recieve the answer
    return self.decode_message(sock.recv())
```

Redundancy's Not Easy

- Several things can only run once (cronjobs)
- We currently have a best-effort distributed locking daemon to help with this

Eventlet Everywhere

What is Eventlet?

- Coroutine-based asynchronous concurrency
- Basically, lightweight threads with explicit context switching
- Reads quite like procedural code

Highly Contrived Example

```
import eventlet
from eventlet.green import urllib2
urls = ['http://ep.io', 'http://t.co']
results = []

def fetch(url):
  results.append(urllib2.urlopen(url).read())

for url in urls:
  eventlet.spawn(fetch, url)
```

Integration

- Most of our codebase uses Eventlet (~20,000 lines)
- Used for concurrency in daemons, workers, and batch processing
- ZeroMQ and Eventlet work together nicely

Why?

- Far less race conditions than threading
- Multiprocessing can't handle ~2000 threads
- More readable code than callback-based systems

The Upload Process

Background

- Every time an app is uploaded to ep.io it gets a fresh app image to deploy into
- Each app image has its own virtualenv
- The typical ep.io app has around 3 or 4 dependencies
- Some have more than 40

Parellised pip

- Installing 40 packages in serial takes quite a while
- Our custom pip version installs them in parallel, with caching
- Not 100% compatable with complex dependency sets yet

Some Rough Numbers

- 15 requirements, some git, some pypi:
  - Traditional: ~300 seconds
  - Parellised, no cache: 30 seconds
  - Parellised, cached: 2 seconds

Compiled Modules

- ep.io app bundles are technically architecture-independent
- All compiled dependencies currently installed as system packages with dual 2.6/2.7 versions
- Will probably move to just bundling .so files too

It's not just uploads

- Upload servers are general SSH endpoint
- Also do rsync, scp, command running
- Commands have semi-custom terminal emulation transported over ZeroMQ
- Hope you never have to use pty, ioctl or fcntl

A Little Snippet

```
old = termios.tcgetattr(fd)

new = old[:]

new[0] &= ~(termios.ISTRIP|termios.INLCR|
  termios.IGNCR|termios.ICRNL|termios.IXON|
  termios.IXANY|termios.IXOFF)

new[2] &= ~(termios.OPOST)

new[3] &= ~(termios.ECHO|termios.ISIG|termios.ICANON|
  termios.ECHOE|termios.ECHOK|termios.ECHONL|
  termios.IEXTEN)

tcsetattr_flags = termios.TCSANOW

if hasattr(termios, 'TCSASOFT'):

    tcsetattr_flags |= termios.TCSASOFT
```

The Joy of Networks

It‘s not just the slow ones

- Any network has a significant slowdown compared to local access
- Locking and concurrent access also an issue
- You can‘t run everything on one machine forever

It's also the slow ones

- Transatlantic latency is around 100ms
- Internal latency on EC2 can peak higher than 10s
- Routing blips can cause very short outages

Heuristics and Optimism

- Sites and servers get a short grace period if they vanish in which to reappear
- Another site instance gets booted if needed – if the old one reappears, it gets killed
- Everything is designed to be run at least twice, so launching more things is not an issue

Security

- We treat our internal network as public
- All messages signed/encrypted
- Firewalling of unnecessary ports
- Separate machines for higher-risk processes

General Challenges

The Stack

Three years ago

- Apache and mod_wsgi
- PostgreSQL 8.x
- Memcached

Today

- Nginx (static files/gzipping)
- Gunicorn (dynamic pages, unix socket best)
- PostgreSQL 9
- Redis
- virtualenv

Higher loads?

- Varnish for site caching
- HAProxy or Nginx for loadbalancing
- Give PostgreSQL more resources

Development and Staging

- No need to run gunicorn/nginx locally
- PostgreSQL 9 still slightly annoying to install
- Redis is very easy to set up
- Staging should be EXACTLY the same as live

Backups and Redundancy

Archives != High Availability

- Your PostgreSQL slave is not a backup
- We back up using multiple formats to diverse locations

It‘s not just disasters

- Many other things other than theft and failure can lose data
- Don‘t back up to the same provider, they can cancel your account…

Keep History

- You may not realise you need backups until the next month
- Take backups before any major change in database or code

Check your backups restore

- Just seeing if they're there isn't good enough
- Try restoring your entire site onto a fresh box

Replication is hard

- PostgreSQL and Redis replication both require your code to be modified a bit
- Django offers some help with database routers
- It‘s also not always necessary, and can cause bugs for your users.

An Easy Start

- Dump your database nightly to a SQL file
- Use rdiff-backup (or similar) to sync that, codebase and uploads to a backup directory
- Also sync offsite – get a VPS with a different provider than your main one
- Make your backup server pull the backups, don't push them to it

Sensible Architecture

Ship long-running tasks off

- Use celery, or your own worker solution
- Even more critical if you have synchronous worker threads in your web app
- Email sending can be very slow

Plan for multiple machines

- That means no SQLite
- Make good use of database transactions
- How are you going to store uploaded files?

Loose Coupling

- Simple, loosely-connected components
- Easier to test and easier to debug
- Enforces some rough interface definitions

Automation

- Use Puppet or Chef along with Fabric
- If you do something more than three times, automate it
- Every time you manually SSH in, a kitten gets extremely worried

War Stories

What happens with a full disk?

- Redis and MongoDB have historically both hated this situation, and lost data
- We had this with Redis – there was more than 10% disk free, but that wasn‘t enough to dump everything into.

Stretching tools

- Our load balancer was initally HAProxy
- It really doesn't like having 3000 backends reloaded every 10 seconds
- Custom eventlet-based loadbalancer was simpler and slightly faster

When Usernames Aren't There

- NFSv4 really, really hates UIDs with no corresponding username
- In fact, git does as well
- Variety of workarounds for different tools

Even stable libraries have bugs

- Incompatability between psycopg2 and greenlets caused interpreter lockups
- Fixed in 2.4.2
- Almost impossible to debug

Awkward Penultimate Slide

- You don't have to be mad to write a distributed process management system, but it helps
- ZeroMQ is really, really nice. Really.
- Eventlet is a very useful concurrency tool
- Every developer should know a little ops
- Automation, consistency and preparation are key

Thank you.

Questions, comments or heckles?

Andrew Godwin

andrew@ep.io / @andrewgodwin